Exercise: 4

Name: Pekka Lehtola

How many tasks did you do: 8

Were the tasks easy, ok, difficult: Ok

Do you need help/comments in any task (if yes, to which ones):

1. Explain the following terms and what they are used for:

a. Inheritance (in object-oriented programming)

- It refers to defining a new class with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class. Derived class inherits methods and attributes from parent/base class
- Example of inheritance:

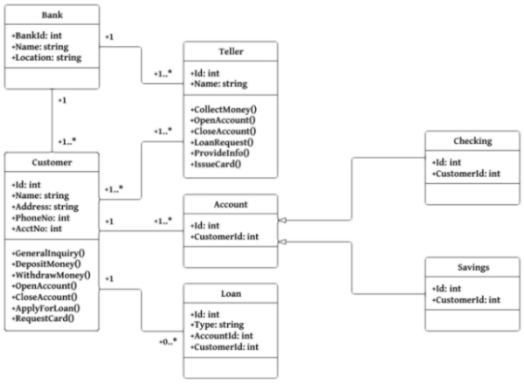
```
class BaseClass:
  Body of base class
class DerivedClass(BaseClass):
  Body of derived class
```

b. UML

- UML or Unified Modeling Language is way of modeling and documenting software.
- It is based on diagrammatic representations of software components. with visual representations, possible flaws or errors in software are detected.

c. UML class diagram

- a class diagram in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
- Example of class diagram:



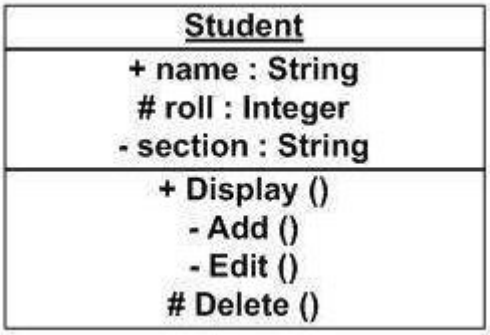
Example of a Class Diagram for a Banking System

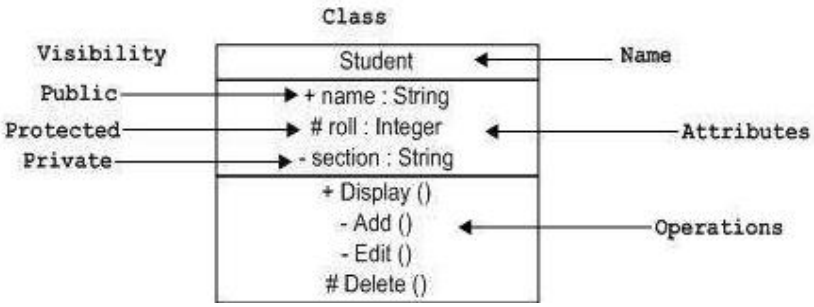
2. Answer the following question.

When you model using UML diagrams, why is it important to folllow the UML syntax strictly?

Efficient and appropriate use of notations is very important for making a complete and meaningful model. The model is useless, unless its purpose is depicted properly.

This model useless with out knowing the correct syntax. UML is very barebones regarding text so every model needs to follow correct syntax to make it readable to everyone.





3. Take the cell phone class of last week and divide the cell phone class into another file (name the file clearly). Leave the main function in the original file. Test, that your code still works.

Screen capture of Task 3

cellphone_main.py

```python
# File name: Cellphone_main
# Author: Pekka Lehtola
# Description: Cellphone class main function

from cellphone_class import Cellphone

def main():

    my_cellphone = Cellphone()

    my_cellphone.set_manufact()
    my_cellphone.set_model()
    my_cellphone.set_retail_price()

    print("Here is the data that you provided :")

    my_cellphone.get_manufact()
    my_cellphone.get_model_number()
    my_cellphone.get_retail_price()

main()
```

cellphone_class.py

```python
# File name: Cellphone class
# Author: Pekka Lehtola
# Description: Cellphone class

class Cellphone:

    def __init__(self):

        self.manufact = ""
        self.model = ""
        self.retail_price = 0

    def set_manufact(self):

        self.manufact = str(input("Enter the manufacturer : "))

    def set_model(self):

        self.model = str(input("Enter the model number : "))

    def set_retail_price(self):

        self.retail_price = float(input("Enter the retail price : "))

    def get_manufact(self):

            return print("Manufacturer:", self.manufact)

    def get_model_number(self):
        return print("Model number:", self.model)

    def get_retail_price(self):
        return print("Retail price:", self.retail_price)
```

Screen capture of the output of Task 3

Run: cellphone_main

```
C:\Users\pekka\AppData\Local\Microsoft
Enter the manufacturer : Samsung
Enter the model number : Galaxy S7
Enter the retail price : 720
Here is the data that you provided :
Manufacturer: Samsung
Model number: Galaxy S7
Retail price: 720.0

Process finished with exit code 0
```

4. Add and ID data attribute (integer between 1-6) to the cell phone. Cell phone class shall have accessor and mutator methods for all data attributes. Also check the __str__ method is up to date.

Screen capture of Task 4

```python
# File name: Cellphone_main
# Author: Pekka Lehtola
# Description: Cellphone class main function

from cellphone_class import Cellphone


def main():

    my_cellphone = Cellphone()

    my_cellphone.set_manufact()
    my_cellphone.set_model()
    my_cellphone.set_retail_price()
    my_cellphone.set_id()

    print("Here is the data that you provided :")

    my_cellphone.get_manufact()
    my_cellphone.get_model_number()
    my_cellphone.get_retail_price()
    my_cellphone.get_id()

    print(my_cellphone)


main()
```

```python
# File name: Cellphone class
# Author: Pekka Lehtola
# Description: Cellphone class

#Added id attribute and made every attribute private.
class Cellphone:

    def __init__(self):

        self.__manufact = "N/A"
        self.__model = "N/A"
        self.__retail_price = 0
        self.__id = 0

    #__str__ method for clear output prints
    def __str__(self):
        return f"""
    Manufacturer: {self.__manufact}
    Model number: {self.__model}
    Retail price: {self.__retail_price}
    Cellphone ID: {self.__id} """

    #Set methods for modifying private attributes
    def set_manufact(self):

        self.__manufact = str(input("Enter the manufacturer : "))

    def set_model(self):

        self.__model = str(input("Enter the model number : "))

    def set_retail_price(self):

        self.__retail_price = float(input("Enter the retail price : "))

    def set_id(self):

        self.__id = int(input("Enter phone id (1 - 6): "))

        #Used for checking if input is valid
        if self.__id >= 7 or self.__id <= 0:
            print("Value error.")
            self.set_id()

    #used for returning private data attributes.
    def get_manufact(self):

        return print("Manufacturer:", self.__manufact)

    def get_model_number(self):
        return print("Model number:", self.__model)

    def get_retail_price(self):
        return print("Retail price:", self.__retail_price)

    def get_id(self):
        return print("Cellphone ID: ", self.__id)
```

Screen capture of the output of Task 4

```
Run:  cellphone_main
C:\Users\pekka\AppData\Local\Microsoft\WindowsApp
Enter the manufacturer : Samsung
Enter the model number : A70
Enter the retail price : 399.99
Enter phone id (1 - 6): 7
Value error.
Enter phone id (1 - 6): 4
Here is the data that you provided :
Manufacturer: Samsung
Model number: A70
Retail price: 399.99
Cellphone ID:  4

    Manufacturer: Samsung
    Model number: A70
    Retail price: 399.99
    Cellphone ID: 4

Process finished with exit code 0
```

I had outputs with accessor and with __str__ method.

5. Create different cell phone objects (which have different data attribute values, use mutator methods to change the data attribute values). Print out each object's state (use the __str__ method in the cell phone class).

Screen capture of Task 5

```python
# File name: Cellphone_main
# Author: Pekka Lehtola
# Description: Cellphone class main function

from cellphone_class import Cellphone

#List of all cellphone objects
cellphone_object_list = []

def create_cellphones():

    global cellphone_object_list
    ID = 0

    #Ask user how many objects are created
    #ID is given automaticly for phone.
    #Lastly add object to list
    for i in range(int(input("How many cellphones you want to create: "))):

        users_cellphone = input("Enter objects name: ")

        users_cellphone = Cellphone()
        users_cellphone.set_manufact()
        users_cellphone.set_model()
        users_cellphone.set_retail_price()

        ID += 1
        users_cellphone.set_id(ID)

        cellphone_object_list.append(users_cellphone)

        print()

#Prints every cellphone using __str__ method
def main():

    create_cellphones()
    print("Here is the cellphones you created: ")

    for object in cellphone_object_list:
        print(object)

main()
```

No changes to Cellphone class.

Screen capture of the output of Task 5

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe
How many cellphones you want to create: 3
Enter objects name: samsung
Enter the manufacturer : Samsung
Enter the model number : A70
Enter the retail price : 450

Enter objects name: oneplus
Enter the manufacturer : Oneplus
Enter the model number : 8T
Enter the retail price : 980.50

Enter objects name: apple
Enter the manufacturer : Apple
Enter the model number : iPhone 8
Enter the retail price : 600

Here is the cellphones you created:

    Manufacturer: Samsung
    Model number: A70
    Retail price: 450.0
    Cellphone ID: 1


    Manufacturer: Oneplus
    Model number: 8T
    Retail price: 980.5
    Cellphone ID: 2


    Manufacturer: Apple
    Model number: iPhone 8
    Retail price: 600.0
    Cellphone ID: 3


Process finished with exit code 0
```

6. Take the Dice class from your earlier exercises and place that to its own file. Then in main function roll a dice and based on the result choose the correct cell phone based on the ID. Print out the chosen cell phone object's state.

Screen capture of Task 6

```python
# File name: Cellphone_main
# Author: Pekka Lehtola
# Description: Cellphone main function, returned cellphone
#              selected with dice roll.

from cellphone_class import Cellphone
from dice_class import Dice

#List of all cellphone objects
cellphone_object_list = []

def create_cellphones():

    global cellphone_object_list
    ID = 0

    #Ask user how many objects are created
    #ID is given automaticly for phone.
    #Lastly add object to list
    for i in range(int(input("How many cellphones you want to create: "))):

        users_cellphone = input("Enter objects name: ")

        users_cellphone = Cellphone()
        users_cellphone.set_manufact()
        users_cellphone.set_model()
        users_cellphone.set_retail_price()

        ID += 1
        users_cellphone.set_id(ID)

        cellphone_object_list.append(users_cellphone)

        print()

#Prints every cellphone using __str__ method
def main():


    create_cellphones()

    #Imported Dice class
    #Dice rolls random number and with that cellphone is selected
    dice = Dice()
    dice.roll_the_dice()
    dice.get_side_up()

    print("Here is the cellphones you created: ")

    #Returns every object in list
    for object in cellphone_object_list:

        print(object)

    print("Here is cellphone selected with dice roll: ")

    #Returns object that has same ID as dices side up value.
    for object in cellphone_object_list:

        if dice.side_up == int(object.get_only_id()):
            print(object)

main()
```

Modified Dice class.

```python
# File name: dice_class
# Author: Pekka Lehtola
# Description: Roll a dice.

import random

class Dice:

    # Dice with three attributes: Color,
    def __init__(self):

        self.color = "White"
        self.side_up = 1
        self.sum_of_throws = 0

    # Selecting random number between 1 and 6, form that selectin predefined side color.
    # Modifying dices values and calculating new sum.
    def roll_the_dice(self):

        random_number = random.randint(1,6)
        colors = {1: "Red", 2: "Blue", 3: "Green", 4: "Yellow", 5: "Black", 6: "White"}

        self.sum_of_throws += random_number
        self.side_up = random_number
        self.color = colors[self.side_up]

        print("Rolling the dice...")

    def get_side_up(self):

        return print("Dices side up is:", self.side_up)

    # Prints objects str method.
    def get_dice_state(self):

        print("Checking dice...")
        print(self)

    # Runs init method again to reset the sum of throws
    def restart_game(self):

        self.__init__()
        print("Restarting the game...", end="\n\n")

    # Defining printing of the object.
    def __str__(self):

        return f"""Dices color is {self.color}, the sideup is {self.side_up}
and sum of throws is {self.sum_of_throws} \n """
```

Added get_only_id to Cellphone class.

```python
    def get_only_id(self):
        return self.__id

    def get_id(self):
        return print("Cellphone ID: ", self.__id)
```

Screen capture of

the output of Task 6

```
Run: cellphone_main (2)
How many cellphones you want to create: 6
Enter objects name: samsung
Enter the manufacturer : Samsung
Enter the model number : Galaxy s2
Enter the retail price : 620

Enter objects name: oneplus
Enter the manufacturer : Oneplus
Enter the model number : 7T
Enter the retail price : 300

Enter objects name: motorola
Enter the manufacturer : Motorola
Enter the model number : Razr
Enter the retail price : 200

Enter objects name: apple
Enter the manufacturer : Apple
Enter the model number : iPhone XR
Enter the retail price : 999

Enter objects name: google
Enter the manufacturer : Google
Enter the model number : Pixel 6
Enter the retail price : 870

Enter objects name: huawei
Enter the manufacturer : Huawei
Enter the model number : Honor 10
Enter the retail price : 555

Rolling the dice...
Dices side up is: 5
Here is the cellphones you created:

    Manufacturer: Samsung
    Model number: Galaxy s2
    Retail price: 620.0
    Cellphone ID: 1


    Manufacturer: Oneplus
    Model number: 7T
    Retail price: 300.0
    Cellphone ID: 2


    Manufacturer: Motorola
    Model number: Razr
    Retail price: 200.0
    Cellphone ID: 3


    Manufacturer: Apple
    Model number: iPhone XR
    Retail price: 999.0
    Cellphone ID: 4


    Manufacturer: Google
    Model number: Pixel 6
    Retail price: 870.0
    Cellphone ID: 5


    Manufacturer: Huawei
    Model number: Honor 10
    Retail price: 555.0
    Cellphone ID: 6

Here is cellphone selected with dice roll:

    Manufacturer: Google
    Model number: Pixel 6
    Retail price: 870.0
    Cellphone ID: 5


Process finished with exit code 0
```

7. Create a car object. It has the following data attributes: make, model, mileage, price, color, maximum load limit, size of trunk. Make them private. Write accessor and mutator methods to change them. Add __str__ method to print the state of the car.

Screen capture of Task 7

```python
# File name: car_class
# Author: Pekka Lehtola
# Description: car_class with private attributes and str method.

class Car:

    def __init__(self):

        self.__make = "make"
        self.__model = "model"
        self.__mileage = "mileage"
        self.__price = "price"
        self.__color = "color"
        self.__maximum_load = "maximum_load"
        self.__trunk_size = "trunk_size"

    #__str__ method for clean printing
    def __str__(self):
        return f"""
        Make: {self.__make}
        Model: {self.__model}
        Mileage: {self.__mileage} Km
        Price: {self.__price} $
        Color: {self.__color}
        Maximum load {self.__maximum_load} Kg
        Size of trunk {self.__trunk_size} m^3
        """

    #All of the set methods.
    def set_make(self):
        self.__make = input("Set make for the car: ")
    def set_model(self):
        self.__model = input("Set model for the car: ")
    def set_milage(self):
        self.__mileage = int(input("Set mileage for the car: "))
    def set_price(self):
        self.__price = float(input("Set price for the car: "))
    def set_color(self):
        self.__color = input("Set color for the car: ")
    def set_maximum_load(self):
        self.__maximum_load = input("Set maximum load for the car: ")
    def set_size_of_trunk(self):
        self.__trunk_size = input("Set the size of the trunk: ")

    #All of the get methods.
    def get_make(self):
        return self.__make
    def get_model(self):
        return self.__model
    def get_milage(self):
        return self.__mileage
    def get_price(self):
        return self.__price
    def get_color(self):
        return self.__color
    def get_maximum_load(self):
        return self.__maximum_load
    def get_size_of_trunk(self):
        return self.__trunk_size
```

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main function for exercise 4_7

from car_class import Car

def main():

    honda = Car()

    honda.set_make()
    honda.set_model()
    honda.set_milage()
    honda.set_price()
    honda.set_color()
    honda.set_maximum_load()
    honda.set_size_of_trunk()

    print(honda)

main()
```

Screen capture of the output of Task 7

```
Run: main
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\
Set make for the car: Honda
Set model for the car: Civic
Set mileage for the car: 90000
Set price for the car: 3500
Set color for the car: Black
Set maximum load for the car: 400
Set the size of the trunk: 2

        Make: Honda
        Model: Civic
        Mileage: 90000 Km
        Price: 3500.0 $
        Color: Black
        Maximum load 400 Kg
        Size of trunk 2 m^3


Process finished with exit code 0
```

Screen capture of Task 8

Mammal class:

```python
# File name: mammal_class
# Author: Pekka Lehtola
# Description: class for creating mammals

class Mammal:

    def __init__(self, ID, species, name, weight, width, breadth, height):

        self.id = ID
        self.species = species
        self.name = name
        self.size = float((width*breadth*height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3
        self.weight = weight
        self.width = width
        self.breadth = breadth
        self.height = height

    #str method for clean output printing with correct units
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size {self.size} m^3
    Weight {self.weight} Kg
    Width {self.width} cm
    Breadth {self.breadth} cm
    Height {self.height} cm
        """
```

Main:

```python
# File name: main
# Author: Pekka Lehtola
# Description: Main function for exercise 4_8

#Required imports
from car_class import Car
from mammal_class import Mammal
from dice_class import Dice

#List for storing mamal objects
mammal_list = []

#Automated mamal object creator
def create_mammals():

    global mammal_list

    ID = [1, 2, 3, 4, 5, 6]
    obj = ["dog", "horse", "cat", "cow", "rabbit", "monkey"]
    species = ["Dog", "Horse", "Cat", "Cow", "Rabbit", "Monkey"]
    name = ["Fluffy", "Linda", "Snuffles", "Mansikki", "Bucks", "George"]
    weight = [60, 400, 10, 700, 3, 30]
    width = [40, 70, 10, 90, 8, 20]
    breadth = [100, 200, 50, 170, 40, 20]
    height = [60, 160, 30, 120, 50, 60]

    #Takes the same index from every list and combines it to one mamal object.
    #And when every mamal created prints them out.
    for i in range(len(ID)):

        obj[i] = Mammal(ID[i], species[i], name[i], weight[i], width[i], breadth[i], height[i])
        mammal_list.append(obj[i])

    for object in mammal_list:
        print(object)
```

```python

#If the animal wasn't too heavy, check if animals dimension are too big for trunk.
def check_size_of_trunk(car, mammal):

    if car.get_size_of_trunk() < float(mammal.size):

        print("Sorry mammal too big to fit into trunk...")
        return print("Trunk size was", car.get_size_of_trunk(), "m^3 and animal was", mammal.size, "m^3")

    else:
        return print("congratulations you managed to fit the mammal into car! ")

#Check if animals weight is bigger than cars maximum weight limit.
def check_car_max_load(car, mammal):

    if car.get_maximum_load() < int(mammal.weight):

        print("Sorry mammal is too heavy...")
        return print("Max weight was: ", car.get_maximum_load(), "Kg and animal weighted", mammal.weight, "Kg")

    else:
        print("Mammal not too heavy. ")
        check_size_of_trunk(car, mammal)

# First main creates all mamals. Then car object is created. Then dice is rolled.
#Dice value is compared to mamals ID values in mammal list. Finally main runs weight and size
#validation.
def main():

    create_mammals()

    car = Car()

    car.set_make()
    car.set_model()
    car.set_milage()
    car.set_price()
    car.set_color()
    car.set_maximum_load()
    car.set_size_of_trunk()

    print(car)

    dice = Dice()
    dice.roll_the_dice()
    dice.get_side_up()

    for object in mammal_list:

        if object.id == dice.side_up:
            print("Selected animal: ")
            print(object)
            check_car_max_load(car, object)

main()

```

Dice and Car class unchanged.

Screen capture of the output of Task 8

```
Set make for the car: Honda
Set model for the car: Civic
Set mileage for the car: 50000
Set price for the car: 12000
Set color for the car: Silver
Set maximum load for the car: 150
Set the size of the trunk: 1.5

        Make: Honda
        Model: Civic
        Mileage: 50000 Km
        Price: 12000.0 $
        Color: Silver
        Maximum load 150 Kg
        Size of trunk 1.5 m^3

Rolling the dice...
Dices side up is: 4
Selected animal:

    ID: 4
    Species: Cow
    Name: Mansikki
    Size 1.836 m^3
    Weight 700 Kg
    Width 90 cm
    Breadth 170 cm
    Height 120 cm

Sorry mammal is too heavy...
Max weight was:  150 Kg and animal weighted 700 Kg

Process finished with exit code 0
```

```
Set make for the car: Chevrolet
Set model for the car: Cruze
Set mileage for the car: 70000
Set price for the car: 12000
Set color for the car: Ice Blue
Set maximum load for the car: 400
Set the size of the trunk: 1.8

        Make: Chevrolet
        Model: Cruze
        Mileage: 70000 Km
        Price: 12000.0 $
        Color: Ice Blue
        Maximum load 400 Kg
        Size of trunk 1.8 m^3

Rolling the dice...
Dices side up is: 2
Selected animal:

    ID: 2
    Species: Horse
    Name: Linda
    Size 2.24 m^3
    Weight 400 Kg
    Width 70 cm
    Breadth 200 cm
    Height 160 cm

Mammal not too heavy.
Sorry mammal too big to fit into trunk...
Trunk size was 1.8 m^3 and animal was 2.24 m^3

Process finished with exit code 0
```

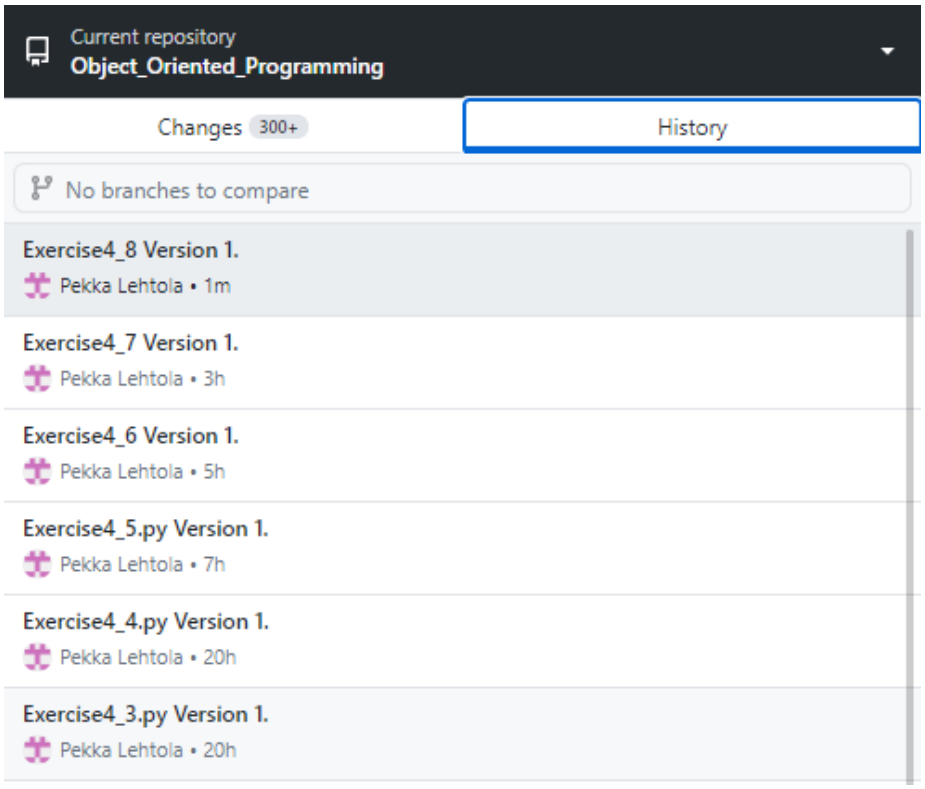
```
Set make for the car: Volvo
Set model for the car: 520
Set mileage for the car: 178000
Set price for the car: 6200
Set color for the car: Black
Set maximum load for the car: 300
Set the size of the trunk: 2

        Make: Volvo
        Model: 520
        Mileage: 178000 Km
        Price: 6200.0 $
        Color: Black
        Maximum load 300 Kg
        Size of trunk 2.0 m^3

Rolling the dice...
Dices side up is: 1
Selected animal:

    ID: 1
    Species: Dog
    Name: Fluffy
    Size 0.24 m^3
    Weight 60 Kg
    Width 40 cm
    Breadth 100 cm
    Height 60 cm

Mammal not too heavy.
congratulations you managed to fit the mammal into car!

Process finished with exit code 0
```

Mammal creating was the same in every instance.

Mammals were printed First.

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsA

    ID: 1
    Species: Dog
    Name: Fluffy
    Size 0.24 m^3
    Weight 60 Kg
    Width 40 cm
    Breadth 100 cm
    Height 60 cm


    ID: 2
    Species: Horse
    Name: Linda
    Size 2.24 m^3
    Weight 400 Kg
    Width 70 cm
    Breadth 200 cm
    Height 160 cm


    ID: 3
    Species: Cat
    Name: Snuffles
    Size 0.015 m^3
    Weight 10 Kg
    Width 10 cm
    Breadth 50 cm
    Height 30 cm


    ID: 4
    Species: Cow
    Name: Mansikki
    Size 1.836 m^3
    Weight 700 Kg
    Width 90 cm
    Breadth 170 cm
    Height 120 cm


    ID: 5
    Species: Rabbit
    Name: Bucks
    Size 0.016 m^3
    Weight 3 Kg
    Width 8 cm
    Breadth 40 cm
    Height 50 cm


    ID: 6
    Species: Monkey
    Name: George
    Size 0.024 m^3
    Weight 30 Kg
    Width 20 cm
    Breadth 20 cm
    Height 60 cm
```

Screen capture of git log (showing that you made a commit after every task).



Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Tällä viikolla tehtävät olivat ihan ok, tietenkin oli joitakin haasteita, mutta vikastakin tehtävästä selvisin ihan hyvin.

Doing this exercise, I learned…

Objectien sijoitusta listaan ja objection kommunikointia toistensa kanssa.

I am still wondering…

Vaikka onnistuin sijoittamaan objecteja listoihin ja käyttämään niitä sieltä, olen aika epävarma kyseisen käytännön kanssa vielä. En tiedä missä muodossa ne tallentuvat sinne ja miten käyttäisin niitä listasta tehokkaasti. En keksinyt esimerkiksi miten kutsuisin niitä listasta objectin nimen mukaan.

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

Onko olemassa järkevämpää tapaa hakea objecti listasta kuin tämä. Esimerkiksi jos olen luonut kolme objectia (Kissa, Koira, Lisko) onko suoraa tapaa kutsua Liskoa listasta ilman for looppia tai käyttämällä indexia missä kohtaa sijaitsee listassa.

```
for object in mammal_list:

    if object.id == dice.side_up:
        print("Selected animal: ")
        print(object)
        check_car_max_load(car, object)
```